George C. Marshall Space Flight Center
Marshall Space Flight Center, Alabama 35812

QD-QE-007
REVISION C
EFFECTIVE DATE: September 24, 2004

ORGANIZATIONAL INSTRUCTION

Software Quality Assurance Planning

OPR(s)	OPR DESIGNEE
QD10,QD20,QD30,QD40	Rosalynne Strickland

DOCUMENT HISTORY LOG

Status (Baseline/ Revision/ Canceled)	Document Revision	Effective Date	Description
Baseline		10/13/00	
Revision	A	9/09/02	Format and numbering change to implement requirements of QS-A-001 rev F.
Revision	B	10/20/03	Changes made to incorporate new QS40 organizational name
Revision	C	9/24/04	Revised to bring document in compliance with the HQ Rules Review Action (CAITS: 04-DA01-0387). Changes were also made to reflect S&MA organizational name changes (i.e., QS to QD).

Software Quality Assurance Planning

1.0 PURPOSE, SCOPE, and APPLICABILITY

1.1 Purpose - The purpose of the instruction is to provide assistance to the Software Assurance representative responsible for SQA planning.

1.2 Scope - This instruction explains and clarifies the contents of each section of the Software Quality Assurance Plan. The SQAP shall describe the plans and activities for the Software Quality Assurance (SQA) staff. The SQA staff observes the development process and reports deficiencies observed in the procedures and the resulting products.

1.3 Applicability - This organizational instruction applies to Software Assurance by QD40.

2.0 DOCUMENTS

2.1 Applicable Documents

2.1.1 QS-QE-009 SOFTWARE ASSURANCE REVIEW/APPROVAL OF TECHNICAL DOCUMENTS

2.1.2 QS-QE-010 SOFTWARE ASSURANCE SOFTWARE MILESTONE REVIEW SUPPORT

2.2 Reference Documents

2.2.1 QS-QE-008 SOFTWARE ASSURANCE STATUS REPORT

3.0 DEFINITIONS

3.1 Acceptance Testing - Testing conducted to determine whether or not a system satisfies its requirements and to determine whether or not to accept the system.

3.2 Baseline - A product that has been formally reviewed and approved that can be changed only through formal change control procedures.

3.3 Change Control - The process, by which a change to a baseline is proposed, evaluated, approved or rejected, scheduled, and tracked.

3.4 Component - A basic useable part of a system or program, an assembly of modules.

3.5 Configuration - A set of software, documentation, and data elements that meets a set of requirements or contractual obligations. The items that make up a baseline.

3.6 Configuration Control - Configuration control is a process to provide the administrative mechanism for precipitating, preparing, evaluation, and approving or disapproving all change proposals throughout the system life cycle. That is, software configuration control is change proposal processing.

3.7 Configuration Identification - Configuration identification includes the specifications and their associated diagrams, flow charts, drawings, parts lists, etc., that are used to describe the functional and physical characteristics of a CI.

3.8 Configuration Status Accounting and Reporting - Software configuration status accounting is the administrative tracking and reporting of all software items formally identified and controlled.

3.9 Debugging - The process of locating, analyzing, and correcting suspected faults in software.

3.10 Design - The process of defining the software architecture, components, modules, interfaces, test approach, and data for a software system.

3.11 Fault - An accidental condition that causes a functional part of a software system to fail to perform a required function or to perform unwanted functions.

3.12 Firmware - Computer programs and data loaded in a type of memory that cannot be dynamically modified (i.e. PROMS, EPROM's).

3.13 Inspection - A formal technique in which SW requirements, design, or code are reviewed in detail by a person or group other than the author to detect faults, violations of standards, and other problems.

3.14 Module - A part of a computer program that is separable and identifiable with respect to compiling, combining with other parts, and loading. A subroutine is an example of a module.

3.15 Nonconformance - Any deviation of hardware, software, or documentation from its functional, performance, or interface requirements or from the standards to which it is to be developed.

3.16 Security - The protection of computer hardware and software from accidental and deliberate unauthorized access, use, modification, destruction, or disclosure.

3.17 Software - Computer programs, procedures, associated documentation and data. At MSFC the term software includes firmware.

3.18 Software Assurance -The planned and systematic set of activities that ensure that software life cycle processes and products conform to requirements, standards, and procedures. Software Assurance includes the disciplines of Software Quality (functions of Software Quality Engineering, Software Quality Assurance, Software Quality Control), Software Safety, Software Reliability, Software Verification and Validation, and Independent Verification and Validation (IV&V).

3.19 The Software Configuration Item - A defined software product that satisfies an end use function and is designated for configuration management.

3.20 Software Configuration Management (SCM) is a discipline applying technical and administrative direction and surveillance to (1) identify and document the functional and physical characteristics of software configuration items and baseline, (2) control changes to those characteristics, and (3) record and report change processing and implementation status of software configuration items and baseline.

3.21 The Software Development Agent (SDA) is the NASA organization or contractor responsible for software management, development, and assurance. For in-house software development, the contract may be the Project Plan, Software Development Plan, or other governing document.

3.22 Software Quality Assurance (SQA) is the planned, systematic process that ensures that desired procedures, standards, requirements, and quality attributes are:

- Established prior to software acquisition/development
- Followed during each phase of acquisition/development

Ultimately, the basic Software Quality Assurance function is to ensure that both software products and acquisition process comply with established standards, practices, and procedures.

3.23 Software Library - A collection of software and related documentation that is designed to aid in software development, use, maintenance, or control.

3.24 System - A collection of software programs organized to accomplish a specific set of functions or to meet a set of requirements.

3.25 Test Plan - A document describing the approach to a planned testing activity. The plan typically identifies the requirements and items to be tested, the test to be run, test schedules, resources and data requirements, reports to be produced, and evaluation criteria.

3.26 Test Procedure - A document giving detailed instructions for the setup, operation, and results for a given test.

3.27 Tool - A program used to assist in the development, testing, analysis, or maintenance of another computer program or its documentation.

3.28 Unit - Separately named and accessible elements of software, which perform specific functions. Also called subroutines, functions, procedures, or modules.

3.29 Unit Development Folder - A formalized set of records documenting the development of a software unit. It includes schedules, reviews, approvals, and supporting documentation.

3.30 Validation - The process of evaluation of software to assure that it meets its requirements. It is normally done by reviews and testing.

3.31 Verification - The process of determining whether the products of a given phase of the software development cycle fulfill the requirements established during the previous phase

4.0 INSTRUCTIONS

4.1 Contents of a Software Quality Assurance Plan - The Software Quality Assurance Plan shall include the sections listed below to be in compliance. The sections should be ordered in the described sequence. If there is no information pertinent to a section, the following shall appear below the section heading, "This section is not applicable to this plan," together with the appropriate reasons for the exclusion.

1) Purpose
2) Reference documents
3) Management
4) Documentation
5) Standards, practices, and conventions
6) Reviews and audits
7) Test
8) Problem reporting and corrective action
9) Tools and techniques
10) Code Control
11) Media Control
12) Contractor Control
13) Records collection, maintenance, and retention
14) Training

Additional sections shall be added as required.
Some of the SQAP required information might be contained in other documents, such as a separate Software Configuration Management Plan (SCMP), Software Development Plan (SDP), Software Management Plan (SMP), or Software Test Plan (STP). The required information also may be documented in approved standards and accepted conventions,

practices, or procedures. The sections of the SQAP shall reference the documents in which the information is contained. The referenced documents shall be reviewed to ensure that they provide all the required information.

4.2 Purpose (Section 1 of the SQAP) - The following questions shall be addressed in this section:

4.2.1 What is the intended use of the software covered by this SQAP? How is the software to be used? How critical is this software? Is it part of a larger system? If so, how is it related to the system?

4.2.2 What is the scope of this SQAP? Who does it apply to?

4.2.3 Why is this SQAP being written? Is this plan being written in response to an internal (e.g., in-house goals) or external (e.g., legal or contractual) requirement?

4.2.4 Which software items are covered by this SQAP? Specific names and abbreviations should be supplied for these items.

4.2.5 Which portions of the software life cycle apply to each of the software items? Name the life cycle model to be used by this project. For projects enhancing, modifying, and correcting existing products, identify the life cycle stages or phases they will pass through before being integrated into the existing product. Depending upon complexity of the relationship of the software items to the portions of the software life cycle, a matrix may be provided or referenced.

4.2.6 Why were the documents that form the basis of this SQAP chosen?

4.3 Reference documents (Section 2 of the SQAP) - Documents used to develop the SQAP shall be referenced in the text of the SQAP (e.g., military, industry-specific, or corporate quality assurance standards and guidelines). Some may be located with the project and some may be located elsewhere. Identify any special arrangement to obtain the document and to ensure the project uses the most current official version.

4.4 Management (Section 3 of the SQAP) - Section 3.3.1 shall describe each major element of the organization that influences the quality of the software. Section 3.3.2 shall list the tasks covered by this plan. Section 3.3.3 shall identify specific organizational responsibilities for each task. Section 3.3.3 shall also identify the management position that retains overall authority and responsibility for software quality.

4.4.1 The organizational element(s) responsible for the software quality assurance functions covered by the SQAP shall be a dedicated quality assurance element serving a number of projects which implements the SQA functional activities. The most effective organization is a separate Quality Assurance (QA) team that is responsible to an SQA organization rather than to

the manager of the software development organization. SQA independence is necessary because the QA manager must not have development responsibilities that tend to override quality concerns. A pictorial organizational structure shall be included with an explanation describing the nature and degree of relationships with all organizational elements responsible for software quality and development. The explanation shall include the following:

4.4.1.1 A description of each element that interacts with the SQA element.

4.4.1.2 The organizational element delegating authority and delegated responsibilities of interacting elements.

4.4.1.3 Reporting relationships among the interacting elements identifying dependence/independence.

4.4.1.4 Identification of the organizational element with product release authority.

4.4.1.5 Identification of the organizational element or elements that approve the SQAP.

4.4.1.6 The reporting lines for escalating conflicts and the method by which conflicts are to be resolved among the elements.

4.4.1.7 The size of the SQA element and the amount of effort dedicated to the project, where the amount is less than 100%.

4.4.1.8 An explanation of any deviations from the organizational structure outlined in existing SQA policies, procedures, or standards. The description of the organizational structure should be complete so that all the tasks addressed in the SQAP can be directly related to a responsible organization.

4.4.2 The SQA tasks consist of planning activities while others, such as reviews and tests, are directed towards the software product. All the tasks in these sections may not be applicable to a specific project, in which event they may be omitted from the project SQAP. Any omissions or deviations shall be explained in the appropriate section of the project SQAP. Any additional tasks shall be included in the appropriate SQAP sections. This section of the SQAP also shall identify the tasks associated with the publication, distribution, maintenance, and implementation of the SQAP. All tasks are SQA tasks and shall be performed by the SQA function. Each task identified shall be defined together with entrance and exit criteria required initiating and terminating the task. The output of each task shall be defined in such a way that its achievement or termination can be objectively determined in a prescribed manner.

4.4.3 If two or more organizational elements share responsibility for a task, their respective responsibilities shall be identified. Describe the procedure for resolving issues between organizational elements sharing responsibilities. The management position accountable for overall software quality shall be identified. This section of the SQAP shall indicate the review

and approval cycle, indicating signature authority as required. It shall designate the personnel and organizational element responsible for distributing the SQAP and describe the methods and responsibilities for the approval, distribution, and incorporation of changes (all changes should follow the approved CM procedures). The size and complexity of the project may require the duplication of this information in other plans, especially in the Software Development Plan and the Software Management Plan. If duplication exists, reference all documents with duplicate data in order to promote ease of maintenance.

4.5 Documentation (Section 4 of the SQAP) - The SQAP shall identify documentation that will be prepared during the development, verification and validation, use, and maintenance of the software. If there is no independent verification and validation, then the quality assurance procedures that are to be used on the project should be identified. Also, all required test documentation should be noted. The SQAP also shall identify the specific reviews, audits, and associated criteria required for each document. If this information is provided in another document, only a reference should be given. Organizational policies, procedures, and standards may determine additional information requirements.

4.5.1 Minimum documentation requirements - The following documents are required for all projects:

a) Software Requirements Specifications (SRS)
b) Software Design Description (SDD)
c) Software Test Plan (STP)
d) Software Test Report (STR)
e) Operator/User documentation
f) Software Configuration Management Plan (SCMP)

These documents are equally usable in in-house developments with an informal contract or with no contract. Other documentation, such as test plans and database design information, should be provided as applicable. The QA activities associated with each document review shall be scheduled in consonance with the development life cycle phases of the project. A brief description of each document follows.

4.5.1.1 The SRS is usually developed from one or more documents, such as a user requirements statement, operational requirements, preliminary hazard analysis, parent or previous SRS reverse engineering documentation, system-level requirements and design documentation, statement of work, or contract. It specifies in detail the requirements as agreed upon by the developer and the requester or user. The SQAP should identify the governing documents and state the precedence in the event of two or more documents containing contradictory requirements. Where the developer produces the SRS, the SQAP should identify what standards or guides apply to the content and format of the SRS. The SRS may contain a list of references that identify working models (such as prototypes or products) that are part of the requirements and need to be included in the design (software and system), development, testing, and operational phases. The SQAP should clearly define the methods to be used by the

SQA organizational element to verify and validate the data in the SRS. The SRS is subject to the Software Requirements Review (SRR) described in QS-QE-010, which identifies the SQA organizational element's QA activities.

4.5.1.2 The Software Design Description (SDD) is a technical description of how the software will meet the requirements set forth in the SRS. Its most important function is to describe a decomposition of the whole system into components (subsystems, segments, etc.) that are complete and well bounded. In addition, it should document the rationale for the more important design decisions in order to facilitate the understanding of the system structure. The SDD describes major system features such as databases diagnostics, external and internal interfaces, as well as the overall structure of the design. It involves descriptions of the operating environment, timing, system throughput, tables, sizing, centralized or distributed processing, extent of parallelism, client/server, reusable objects library, program design language (PDL), prototypes, modeling, simulation, etc. The SQAP should identify the standards and conventions that apply to the content and format of the SDD, as well as the process and procedures to be used in developing the SDD. If prototyping, modeling, or simulations are used, the SQA organizational element could observe a demonstration, which is a more efficient way to review and assess written design documentation. Where separate teams produce the SDD, the procedures shall relate to the responsibilities in 3.3.3. The SDD may be an evolving document or a document that is baselined after each significant review. A new version containing a more detailed design description is developed for each subsequent review. The SQAP should identify the number and purpose of the SDD documents. The SDD is subject to the Preliminary Design Review (PDR) and the Critical Design Review (CDR), described QS-QE-010, respectively, which identify the SQA organizational element's QA activities. The SQAP should clearly define the methods to be used by the SQA organizational element to verify and validate the data in the SDD.

4.5.1.3 The Software Test Plan (STP) describes the overall plan for the verification and validation of the software and could be produced and reviewed incrementally. The tasks, methods, and criteria for verification and validation are described. This section should explain the scope of validation testing to ensure the baselined requirements and explain the stages of development that require review and the extent of the verification that will precede such a review. The STP specifies minimum test documentation requirements. The SQAP should identify which standards and conventions apply to the content and format of the STP.

4.5.1.4 The Software Test Report (STR) summarizes the observed status of the software as a result of the execution of the STP. The STR should include the following information:

4.5.1.4.1 Summary of all life cycle V&V tasks.

4.5.1.4.2 Summary of task results.

4.5.1.4.3 Summary of anomalies and resolutions.

4.5.1.4.4 Assessment of overall software quality.

4.5.1.4.5 Summary from the verification matrix.

4.5.1.4.6 Recommendations such as whether the software is, or is not, ready for operational use. The report may be a full report or a subset.

4.5.1.5 Operator/User documentation - The user documentation section of the SQAP (e.g., documents, videotapes, on-line graphic storyboards, and tutorials) should describe the software's operational use and be comprised of the following items:

4.5.1.5.1 User instructions that contain an introduction, a description of the user's interaction with the system (in the user's terminology), and a description of any required training for using the system.

4.5.1.5.2 An overview of the system, its purpose, and description.

4.5.1.5.3 Input/output specifications.

4.5.1.5.4 Samples of original source documents and examples of all input format (forms or displays).

4.5.1.5.5 Samples of all outputs (forms, reports, or displays).

4.5.1.5.6 Instructions for data preparation, data keying, data verification, data proofing, and error correction. Wherever software is capable of damage to user assets (e.g., data base contents) the user should be forewarned to avoid accidental damage.

4.5.1.5.7 References to all documents or manuals intended for use by the users.

4.5.1.5.8 A description of the system's limitations.

4.5.1.5.9 A description of all the error messages that may be encountered, together with recommended steps to recover from each error.

4.5.1.5.10 Procedures for reporting and handling problems encountered during the use of a software item.

4.5.1.5.11 User administration activities (backup, recovery, batch initiation, access control).

4.5.1.6 Software Configuration Management Plan (SCMP) - The SCMP should describe the tasks, methodology, and tools required assuring that adequate Software Configuration Management (SCM) procedures and controls are documented and are being implemented correctly. It is suggested that the CM organizational element prepare the SCMP so that the

SQA organizational element can maintain its independence for CM evaluation purposes. The SQAP should define the extent to which the project requires configuration management. The SCMP should describe the methods to be used for:

4.5.1.6.1 Identifying the software configuration items.

4.5.1.6.2 Controlling and implementing changes.

4.5.1.6.3 Recording and reporting change and problem report implementation status.

4.5.1.6.4 Conducting configuration audits.

4.5.1.6.5 Identifying review and approval cycle as well as signature authority.

4.5.1.6.6 Identifying the personnel responsible for maintaining the baselines and distributing the SCMP. The SQAP shall clearly define the methods to be used by the SQA organizational element to verify and validate the data in the SCMP.

4.5.2 Other documentation:

4.5.2.1 Software Development Plan (SDP) - The SDP can be used as the highest-level planning document governing a project, or could be subordinate within a larger set of plans. The SDP should identify all technical and managerial activities associated with software development. The SDP should specify the following items, which should be reviewed and assessed by the SQA organizational element:

4.5.2.1.1 Description of software development.

4.5.2.1.2 Software development organization responsibilities and interfaces.

4.5.2.1.3 Process for managing the software development (including allocation of resources, project control mechanisms, software metrics, and risk management).

4.5.2.1.4 Technical methods, tools, and techniques to be used in support of the software development.

4.5.2.1.5 Assignment of responsibility for each software development activity.

4.5.2.1.6 Schedule and interrelationships among activities.

4.5.2.1.7 Formal qualification testing organization and approach.

4.5.2.1.8 Software product evaluation during each life cycle phase including subcontractor products. The SQAP shall define the procedures for creating the data in the SDP and criteria for

updating and assuring its quality. Any deviations from the plan should be reconciled between the SQA staff and the software development manager. The plan should be updated and the latest version clearly identified. Obsolete versions should be maintained according to the organization's policy. Procedures should be defined for the approval and distribution of the SDP.

4.5.2.2 Standards and Procedures Manual (SPM) - The SPM should provide details on all project standards and procedures to be followed. The SQAP shall clearly define the methods to be used by the SQA organizational element to verify and validate the data in the SPM.

4.5.2.3 Software Management Plan (SMP) - The SMP can be used in place of an SDP, or as a plan that governs a larger project that has subordinate projects each covered by SDPs. The SMP should identify all technical and managerial activities associated with an instance of software development. The SMP should specify the following items, which should be reviewed and assessed by the SQA organizational element:

4.5.2.3.1 Description of software development (high-level description if details are contained in an SDP. If there is no SDP, full details are required in the SMP).

4.5.2.3.2 Software development and management organizations responsibilities and interfaces.

4.5.2.3.3 Process for managing the software development (including allocation of resources, project control mechanisms, software metrics, and risk management).

4.5.2.3.4 Technical methods, tools, and techniques to be used in support of the software development (high-level description if details are provided in SDP).

4.5.2.3.5 Assignment of responsibility for each activity (e.g., specific skill assignments, key personnel).

4.5.2.3.6 Schedule and interrelationships among activities.

4.5.2.3.7 A list of deliverables. Subcontractor(s) project management plan any deviations from the plan should be reconciled between the SQA staff and the software project manager. The plan should be updated and the latest version clearly identified. Obsolete versions should be maintained according to the organization's policy. Procedures should be defined for the approval and distribution of the SMP.

4.5.3 Additional suggested documentation - The attributes, context, and environment of the product could dictate inclusion of additional documents, such as, but not limited to, the following:

a) User requirements statement
b) External interface specifications

c) Internal interface specifications
d) Operations manual
e) Installation manual
f) Training manual
g) Software security plan

The SQAP shall clearly define the methods to be used by the SQA organizational element to verify and validate the data in the documents.

4.5.3.1 User requirements statement - A user requirements statement can be used as a high-level document preceding the approved SRS for a large development, in place of an SRS in cases where minor changes are made to an operational system that has no SRS, or as a means of passing requirements on to a contractor. The user requirements statement should include, but is not limited to:

4.5.3.1.1 A natural language service request that contains the identity of the requester, the software item name and title, the date the software item was requested and is required, a description of what the software item should do, an abstract of the need for the software item, privacy or security considerations, and a list of potential users of the software item.

4.5.3.1.2 A list of the objectives that are to be satisfied by the software item, as well as any other needs (e.g., administrative, timing, quality) and restraints the user perceives as necessary.

4.5.3.1.3 References to and summarized conclusions from any studies done to define resource requirements (e.g., hardware, software, personnel, plant and facilities, or environmental), feasibility, or cost-benefit analyses. The SQAP should clearly define the methods to be used by the SQA organizational element to verify and validate the data in the user requirement statement.

4.5.3.2 External interface specifications - External interface specifications should be contained within the software requirement specifications, the system design document, or an Interface Control Document (ICD). In situations where the detailed external interface specifications are not available in the design documentation or ICD, a separate external interface specifications document may be required that would provide lower-level detail. The external interface specifications should contain information about files and other connections, such as messages passed, to software items outside the system under development. Consideration should be given to human interfaces, hardware interfaces, environmental constraints (such as weather conditions) and data structures and files or transactions coming from or going to other systems, standards, protocols, timing issues, and throughput or capacity of the interfaces.

4.5.3.3 Internal interface specifications - Internal interface specifications are a subset of the design documentation and may be traced to a software requirement identified in the SRS. In situations where the internal interface specifications are not available in the design

documentation, and/or an ICD, a separate internal interface specification document may be required. The internal interface specifications should contain information about files and other connections among all the components within the system. Consideration should be given to such subjects as transfer of control between modules, passing of data between modules, physical and logical interfaces, common data, timing, and concurrence management techniques.

4.5.3.4 Operations manual - The operations manual should contain at least the following items:

4.5.3.4.1 Operating instructions that include:

1) An introduction
2) Run schedules
3) Setup requirements/procedures
4) Run control procedures
5) Error procedures
6) Security procedures
7) Distribution procedures
8) Backup and recovery procedures
9) Restart procedures
10) Termination procedures
11) Tutorial and practice procedures

4.5.3.4.2 Specifications for the system, including environmental requirements

4.5.3.4.3 Input/output specifications

4.5.3.4.4 Auditing controls

4.5.3.5 Installation manual - An installation manual should contain instructions for the installation of the software, for example, file con-version instructions, use of user-controlled installation options, and instructions for performing an installation test. Installation procedures may be performed through an interactive interface (i.e., menu driven).

4.5.3.6 Training manual - The training manual should contain information necessary for training users and operators of the system. It should contain, but is not limited to, the following:

4.5.3.6.1 An introduction

4.5.3.6.2 How to use the system

4.5.3.6.3 How to prepare input

4.5.3.6.4 Data input descriptions

4.5.3.6.5 Data control descriptions

4.5.3.6.6 How to run the system

4.5.3.6.7 Output distributions

4.5.3.6.8 Description of output data and interpretations (e.g., error messages)

4.5.3.6.9 Tutorial and practice exercises

4.5.3.6.10 How to get help

4.5.3.7 Software security plan - The software security plan should address the way in which the software and the data will be protected from unauthorized access or damage. The software security plan should contain information on the following:

4.5.3.7.1 How the data should be classified and how this will be communicated (e.g., "no trespassing" messages).

4.5.3.7.2 How the users of the software access the application and how that accesses is to be controlled.

4.5.3.7.3 Network design.

4.5.3.7.4 User identifications, passwords, security logging, and auditing.

4.5.3.7.5 Super-user password control and protection of path through the system administrator.

4.5.3.7.6 Physical security.

4.5.3.7.7 Virus protection.

4.5.3.7.8 How employees will be trained on security procedures and practices.

4.5.3.7.9 The method by which this security plans will be audited for compliance.

4.5.3.7.10 Disaster plan.

4.5.3.7.11 Whether the system provides file encryption and decryption capability.

4.6 Standards, practices, and conventions (Section 5 of the SQAP) - This section of the SQAP shall identify the standards (mandatory requirements), practices (recommended approach), and

conventions (accepted guidelines), to be employed by all associated with the project, including management and contractors. It shall specify the phases of the life cycle to which they apply. Also, it shall specify how compliance will be monitored and assured. The SQAP shall identify or reference the standards, practices, and conventions to be used on the project. As a minimum, the information in the following life-cycle phases shall be addressed: software requirements, design, implementation, testing, and maintenance. In addition, the standards, practices, and conventions pertaining to software documentation shall be addressed. The descriptions of the standards, practices, and conventions, are often given in a standards and procedures manual or in an SDP. The SQAP shall clearly define the methods to be used by the SQA organizational element to verify and validate the data.

4.6.1 Requirements phase - Identify or reference the standards, practices, and conventions to be used during the requirements phase. Cite any internal (e.g., project) or external (e.g., military or contractual) standards with which requirements baselining and traceability must comply. Use formal requirements statement languages, either textual or graphic, whenever possible. Provision should be made for a scheme that uniquely identifies each requirement. This facilitates traceability and compliance during the subsequent phases.

4.6.2 Design phase - Identify or reference the standards, practices, and conventions to be used during the preliminary design phase where the overall structure of the software system is defined. Cite any internal (e.g., project) or external (e.g., military or contractual) standards with which the design baselining must comply. Top-down design, which is the most acceptable methodology, should be used whenever feasible. Use of graphic techniques and computer-aided software engineering (CASE) tools that aid in compliance verification should be considered. Object-oriented design methodology should be used with appropriately sized hardware and software in those situations where a simpler design is used. For the detailed design phase, state what standards, practices, and conventions will be used for specifying the internal structure of each program module and the interfaces among them. Address such matters as naming conventions and argument list standards. Give serious consideration to requiring the use of a program design language.

4.6.3 Implementation phase - Identify or reference the standards, practice, conventions, and metrics to be used during the implementation phase. Cite any internal (e.g., project) or external (e.g., military or contractual) standards with which implementation procedures must comply. Address such topics as the end-use computer, programming language(s), module size, declaration statement conventions, naming and labeling conventions, component layout standards, the use of structured coding techniques (or structuring precompilers) and CASE tools. Consider data conversion techniques for new systems that are replacing old ones. Standards for the inclusion of comment statement should be covered here. Use standard support software and software tools whenever possible or state reasons for the use of nonstandard support software and tools.

4.6.4 Test phase - Identify or reference the standards, practices, and conventions to be used during the testing phase. Cite any internal (e.g., project) or external (e.g., military or

contractual) standards with which all levels of testing must comply. This includes unit, integration, system and acceptance testing, as well as regression testing. Identify the test environment, which should be the same as the targeted environment. Address criteria for test repeatability and test coverage such as testing every requirement. Specify techniques for tracing the test coverage to the test set. Indicate whether any support software will be required, and state how and from where this software will be obtained. State the method and process for certification of test tools or other support software used for demonstration.

4.6.5 Maintenance phase - Identify or reference the standards, practices, and conventions to be used to maintain the software. Cite any external (e.g., user, and customer) requirements or standards with which maintenance practices must comply. Cite any internal standards or conventions, such as those describing design, implementation, test, and documentation requirements that affect the maintenance process. Specify the methods and techniques (both manual and automated) for controlling and managing the software maintenance process such as requiring formal written change requests, review and evaluation of change requests, prioritizing and scheduling approved change requests, monitoring of the maintenance process through reviews and audits, and the collection and analysis of data affecting and resulting from the maintenance process.

4.6.6 Documentation Standards - Identify or reference the standards, practices, and conventions to be used in preparing and submitting software documentation. Cite any internal (e.g., project) or external (e.g., military, contractual) standards with which documentation must comply. Cite any document interchange (softcopy) standards that are applicable. The types of documentation that should be addressed may include software plans (e.g., SQAP, SCMP), software development documents (e.g., SRS, SDD), and any software deliverables (e.g., source code, users' manuals, tools).

4.7 Reviews and audits (Section 6 of the SQAP) - The software items produced during the software life cycle process shall be reviewed and audited on a planned basis to determine the extent of progress and to evaluate the technical adequacy of the work and its conformance to software requirements and standards. Technical reviews and audits shall be conducted to evaluate the status and quality of the software development effort. Completion of reviews provides assurance that design integrity is maintained, technical deficiencies are identified, and necessary changes have been identified and implemented. This section of the SQAP shall identify the specific technical and managerial reviews and audits to be conducted with respect to the software development plans, schedules, and environment. It shall describe the procedures to be used in the conduct of reviews and audits, and it shall identify the participants (including contractors) and their specific responsibilities. These review and audit procedures shall identify specific responsibility for the preparation of a report upon the completion of each review. This section shall identify by position or job title that is to prepare these reports, the report format, which is to receive the reports, and associated management responsibilities. The review and audit procedures shall describe the follow-up actions to assure that the recommendations made during the reviews and audits are properly implemented. Also, it shall identify those responsible for performing follow-up actions.

4.7.1 Minimum requirements

a) Software Requirements Review (SRR)
b) Preliminary Design Review (PDR)
c) Critical Design Review (CDR)
d) Software Test Plan Review (STPR)
e) Functional audit
f) Physical audit

Tailoring or inclusion of additional reviews and audits should be made as local, contractual, or project specific conditions dictate.

4.7.1.1 Software Requirements Review (SRR) - The SRR is an evaluation of the Software Requirements Specifications (SRS). The SRR is conducted to ensure the adequacy, technical feasibility, and completeness of the requirements stated in the SRS. The SRR should evaluate the SRS for the attributes required by QD-QE-009 and QD-QE-010 (unambiguous, complete, verifiable, consistent, modifiable, traceable, and usable during the operation and maintenance phase). The review ensures that sufficient detail is available to complete the software design. The SQAP should indicate the organizational element responsible for conducting the SRR. All organizational elements that contribute or are impacted by the requirements should participate. These may include software design, software test, software quality assurance, system engineering, customers, users marketing, manufacturing, security, etc. The SQAP should specify how, during the SRR, the quality of the following items would be assessed:

4.7.1.1.1 Traceability and completeness of the requirement from the next higher-level specification (such as a system specification or user requirements specification).

4.7.1.1.2 Adequacy of rationale for any derived requirements.

4.7.1.1.3 Adequacy and completeness of algorithms and equations.

4.7.1.1.4 Correctness and suitability of logic descriptions that may be warranted.

4.7.1.1.5 Compatibility of external (hardware and software) interfaces.

4.7.1.1.6 Adequacy of the description of and approach to the human-machine interface.

4.7.1.1.7 Consistency in the use of symbols and in the specification of all interfaces.

4.7.1.1.8 Availability of constants and tables for calculations.

4.7.1.1.9 Testability of the software requirements.

4.7.1.1.10 Adequacy and completeness of the verification and acceptance requirements.

4.7.1.1.11 Completeness and compatibility of interface specification and control documentation.

4.7.1.1.12 Freedom from unwarranted design detail.

4.7.1.1.13 Trade-off and design studies that have applicability for decisions on:

a) Data base design and/or real-time design issues.
b) Programming language characteristics and usage.
c) Resource allocation (e.g., storage, machine cycles, I/O channel personnel and hardware).
d) Operating system or executive design, or both.
e) Development versus COTS solution.

4.7.1.1.14 The general description of the size and operating characteristics of all support software (e.g., operational program, maintenance and diagnostic programs, compilers, etc.).

4.7.1.1.15 A description of requirements for the operation of the software and identification of functional requirements such as functional simulation, performance, environmental recording and analysis, exercise configuration, etc. The results of the review should be documented in an SQA Status Report that identifies all deficiencies described in the review and provides a plan and schedule for corrective action.

4.7.1.2 Preliminary Design Review (PDR) - The PDR, the pivotal step in the design phase, is held to evaluate the technical adequacy of the preliminary design, to include architectural design, before the beginning of detailed design. The review assesses the progress, consistency, and technical adequacy of the selected design approach; checks the compatibility of the design with the functional and performance requirements of the SRS; and verifies the existence compatibility of the interfaces between the software, hardware, and end users. The PDR also is conducted to determine that the preliminary SDD defines a suitable software design that fulfills the requirements contained in the SRS. The SQAP shall indicate the organizational element responsible for conducting the PDR. All organizational elements that impose requirements or are impacted by the design shall participate in the review. The SQAP shall specify how, during the PDR, the quality of the following items would be assessed:

4.7.1.2.1 All detailed functional interfaces with other software, system equipment, communication systems, etc., for adequate identification of interface design and design solution adequacy.

4.7.1.2.2 The software design as a whole, emphasizing allocation of software components to functions, functional flows, storage requirements and allocations, software operating sequences, and the design of the database.

4.7.1.2.3 The human factor requirements and the human-machine interfaces for adequacy and consistency of design.

4.7.1.2.4 Testability of the design, such as the existence of data stores and process that supports behavior and state determination.

4.7.1.2.5 Test concepts, requirements, documentation, and tools, for adequacy.

4.7.1.2.6 An analysis of the design for compatibility with critical system timing requirements, estimated running times, absence of race conditions and deadlock states, and other performance requirements.

4.7.1.2.7 Technical accuracy and currency of all available test documentation and its compatibility with the test requirements of the SRS. The results should be documented in a SQA Status Report that identifies all deficiencies discovered during the review. The updated preliminary SDD document should be placed under configuration control to establish the baseline for the detailed software design effort.

4.7.1.3 Critical Design Review (CDR) - The CDR is an evaluation of the completed Software Design Description (SDD), which includes all PDR updates and should provide the low-level details of the design. The CDR is conducted to evaluate the technical adequacy, completeness, and correctness of the detailed design of the software before the start of formal coding (not to include code associated with proof of concept or rapid prototyping activities) or as referenced to the life cycle described in an SDP. The purpose of the CDR is to evaluate the acceptability of the detailed design, to establish that the detailed design satisfies the requirements of the preliminary SDD and the SRS, to review compatibility with the other software and hardware with which the product is required to interact, and to assess the technical, cost, and schedule risks of the product design. The SQAP should indicate the organizational element responsible for conducting the CDR. All other organizational elements that impose requirements or are impacted by the design should participate. The SQAP should specify how, during the CDR, the quality of the following items would be assessed:

4.7.1.3.1 The compatibility of the detailed design with the SRS.

4.7.1.3.2 Available data in the forms of logic diagrams, algorithms, storage, allocation charts, and detailed design representation (e.g., flow charts, program design language) to establish design integrity.

4.7.1.3.3 Compatibility and completeness of interface requirements.

4.7.1.3.4 All external and internal interfaces, including interactions with the database.

4.7.1.3.5 Technical accuracy and currency of all available test documentation and its compatibility with the test requirements of the SRS.

4.7.1.3.6 The requirements for the support and test software and hardware to be used in the development of the product.

4.7.1.3.7 The final design, including function flow, timing, sizing, storage requirements, memory maps, database, and other performance factors.

4.7.1.3.8 Final report of any trade-off analysis and design studies.

4.7.1.3.9 Chosen or recommended design tools and design. The results of the review should be documented in a SQA Status report that identifies all deficiencies discovered during the review and a plan and schedule for corrective actions. The updated SDD document, when placed under configuration control, establishes the baseline for coding.

4.7.1.4 Functional audit - The functional audit compares the software as built (including its executable forms and available documentation) with the software requirements as stated in the baselined SRS. Its purpose is to assure that the code addressed all, and only, the documented requirements and functional capabilities stated in the SRS. The SQAP shall indicate the organizational element responsible for the functional audit. The results are to be documented in the functional audit report, which identifies all discrepancies found, and a plan and schedule for their resolution. Audits are not considered to be complete until all discrepancies have been resolved. Input to the functional audit should consist of the following:

4.7.1.4.1 Software Requirements Specifications (SRS).

4.7.1.4.2 Software Test Report (STR).

4.7.1.4.3 Software Verification and Validation Plan Review (STPR) report.

4.7.1.5 Physical audit - The physical audit compares the code with its supporting documentation. Its purpose is to assure that the documentation to be delivered is consistent and correctly describes the code. The SQAP shall indicate the organizational element responsible for conducting the physical audit. The results of the physical audit are to be documented in the physical audit report, which identifies all discrepancies and the plans for their disposition. Once the plans have been approved and implemented (all dispositions have been completed), the software can be delivered. Input to the physical audit shall consist of the following:

4.7.1.5.1 Software Design Description (SDD).

4.7.1.5.2 Software products (e.g., code, algorithms, graphical user interfaces).

4.7.1.5.3 Associated documentation (e.g., engineering notes, software development files, user documentation).

4.8 Test (Section 7 of the SQAP) - The SQAP shall include the specific software tests and testing not addressed in the STP or other test documentation. The SQAP shall identify and describe the methods, approaches, and techniques to be used (or reference appropriate test documentation that has been developed for the project). This section of the SQAP shall include a description of the test planning, test execution, and evaluation of the results of testing, or include a reference to the appropriate section of the SQAP or to standards and procedures that describe the testing process. The testing process described should adequately address the preparation and review of documentation associated with the testing process, including test plans, test design and test case specifications, test procedures and test instructions, test schedule, test reports, and test incidents and their resolution. Test documentation not specifically referenced in the SQAP shall be delineated, or a reference to the appropriate section of the SQAP or to test documentation standards, procedures, or conventions that describe the test documentation to be provided, should be included in this section. This section shall include a description of the management and organizational responsibilities associated with the implementation of the tests described in this section, or a reference to the appropriate section of the SQAP or standards, procedures, or conventions that describe the management and organizational responsibilities. This section shall include a description of the computer-aided software engineering (CASE) tools used in the support of test activities or a reference to the appropriate section of the SQAP. The SQAP shall clearly define the methods to be used by the SQA organizational element to verify and validate the test plans, test data, and test activities.

4.9 Problem reporting and corrective action (Section 8 of the SQAP) - Problems encountered during software development or operation may result from defects in the software, supporting and development processes, hardware, or operations. Because of a problem and the appropriate corrective action requires a centrally controlled system for monitoring problems and determining root causes. The purposes of problem reporting and corrective action systems are to:

4.9.1 Assure that problems are documented, corrected, and used for process improvement.

4.9.2 Assure that problem reports are assessed for their validity.

4.9.3 Assume reported problems and their associated corrective actions are implemented.

4.9.4 Provide feedback to the developer and the user of problem status. These goals should be satisfied by the problem reporting and corrective action system described in the SQAP. The SQAP shall include methods to be used to assure the reported problems are being properly addressed. The SQAP shall describe the organizational element(s), provisions, and timely procedures for documenting, validating, tracking, and reporting the status of problems and the appropriate corrective action. Validating, tracking, and resolving problems require the coordination of various groups within the organization. The SQAP shall specify the groups responsible for authorizing and implementing problem reporting and corrective actions, and submission of unresolved issues to management for resolution. Also, it shall identify the point in the development process where generation of problem reports is to begin for each class of

development result (e.g., plans, SRS, SDD, code, test documentation). The SQAP shall clearly define the methods to be used by the SQA organizational element to verify and validate the use of problem reporting and corrective action practices and procedures.

4.10 Tools, and techniques (Section 9 of the SQAP) - The SQAP shall identify the tools and techniques, to be used to support software quality assurance. It shall list or reference those tools and techniques, that are available, and those that need to be acquired or developed. The responsible organization(s) also shall be identified.

4.10.1 Tools - SQA software tools aid in the evaluation or improvement of software quality. Typical tools include, but are not limited to, documentation aids, checklists, structuring preprocessors, software development folders/files, nonconformance databases, and software traceability matrices.

4.10.2 Techniques - SQA techniques are technical and managerial procedures that aid in the evaluation and improvement of software quality. Such techniques include review of the use of standards, software inspections, requirements tracing, requirements and design verification.

4.11 Code control (Section 10 of the SQAP) - Code control can be interpreted as the ways and means necessary to protect or ensure the validity of completed code. Once an appropriate baseline has been established, the code should be placed under configuration management in a computer program library. The SQAP shall specify controls (or reference the CM procedures for software change) and a security measure for software change and for protection from inadvertent alteration after the code has been baselined. It shall also define or reference the CM procedures (e.g., see SCMP or the CM section in the SDP) and organizational responsibility for controlling the developed code. The SQAP shall specify or reference a code control procedure that:

4.11.1 Defines the specific software to be controlled.

4.11.2 Describes a standard method for identifying, labeling, and cataloging the software.

4.11.3 Lists the physical location of the software under control.

4.11.4 Describes the location, maintenance, and use of all backup copies.

4.11.5 Describes procedures for distributing a copy.

4.11.6 Identifies the documentation that is affected by changes.

4.11.7 Describes procedures for implementing a new version.

4.11.8 Describes how compliance with the above is assured.

4.12 Media control (Section 11 of the SQAP) - Computer program media can be defined as those media on which computer data are stored. Typically, the storage media are CD-ROM, RAM, disks, or tapes, but could include cards, diskettes, listings, or other forms on which the data reside. The media control methods and facilities shall ensure that:

a) The software is stored and retrieval is assured.

b) Off-site storage and retrieval are provided for critical software and copies of baselined code.

c) The software is accessible only to those with the need of access.

d) The environment is controlled so that the physical media on which the software is stored does not degrade.

e) A description is provided of how compliance with the above is assured.

The SQAP shallreference (e.g., see SCMP or CM section of the SDP) or specify procedures and practices that pertain to the above items. For example, a backup procedure for software could indicate the schedule for backup, the type of media on which it will be placed, the location of the storage, the environment of the storage area, and the method to retrieve the backed-up software. A security system may be in place that allows access to software only through an authorization process. The Software Security Plan should be referenced. The SQAP shall delineate the organization elements responsible for administering and reviewing media control methods and facilities. The method for identifying, labeling, and data logging may be the same in both code and media control.

4.12.1 Unauthorized access - Several methods are available that will provide adequate protection from unauthorized access of computer program media. The primary method is to provide a permanent labeling or identification scheme within the storage media. When a disk or tape is used on a computer, this technique can provide adequate password control or access protection. Other methods include a limited access program library, encryption, external markings, and proprietary statements identifying a controlled program. The physical security of all media also must be considered. SQA activities to verify appropriateness and implementation of access procedures shall be documented in the SQAP. Areas of concern include identifying the programs requiring limited access, adherence to label and file restrictions, ensuring use of adequate external labeling restrictions, and providing a controlled environment such as a program library.

4.12.2 Inadvertent damage or degradation –Providing adequate configuration management techniques, safe storage locations such as fireproof vaults, and packaging practices that are antistatic in design can minimize damage or degradation of the media. Periodic review to ensure use of controlled environmental and cataloging practices will minimize degradation of the media. SQA activities to verify appropriateness and implementation of procedures to

minimize media damage or degradation shall be documented in the SQAP.

4.13 Supplier control (Section 12 of the SQAP) - This section of the SQAP shall specify:

4.13.1 The involvement with the contractor's SQA program.

4.13.2 The procedures for auditing the contractor's conformance to contractually standard, and the contractor's SQAP (an option could be to provide for an independent auditor).

4.13.3 The actions available should the contractor not be in conformance with the contractor's SQAP.

4.14 Records collection, maintenance, and retention (Section 13 of the SQAP) – SQAP shall address records collection, maintenance, and retention procedures.

4.14.1 Records collection - The type of records to be collected, whether softcopy or hardcopy, is determined by the overall objectives for record keeping. These objectives should be documented in the SQAP. Possible objectives are:

4.14.1.1 To provide contractual evidence that the software development process was performed in conformance with established practice:

4.14.1.1.1 The SQAP is being followed and conforms to the requirements of applicable standards.

4.14.1.1.2 The software meets design intent and satisfies contractual requirements.

4.14.1.1.3 Corrective action is effective, timely, and complete (i.e., action items are tracked until resolution).

4.14.1.1.4 Testing has been performed in accordance with test plans.

4.14.1.2 To provide historical or reference data that could be used to discover long-term trends in the organization's development techniques. The documents collected for historical or reference purposes should be capable of providing data for productivity, quality, and methodology studies. The documents should provide sufficient design, implementation, and testing data so as to be useful for future development. In addition to SQA documents, records should include program media containing the exact version of programs and materials used in performing tests to assure, test repeatability at any time in the future.

4.14.2 Records maintenance - The SQAP shall specify the manner in which records will be kept, that is, softcopy, hardcopy, etc. Also, it shall state how records would be stored to protect them from fire, theft, or environmental deterioration. The SQAP shall provide for historical archiving if applicable.

4.14.3 Records retention - The SQAP shall specify the length of retention for each type of record maintained. It is important to state in the SQAP when records should be retained and when they should be destroyed. Length of retention shall comply with appropriate program/project requirements.

4.15 Training (Section 14 of the SQAP) – The SQAP shall address training of personnel designated to perform SQA activities.

5.0 NOTES (References)

None

6.0 SAFETY PRECAUTIONS AND WARNING NOTES

None

7.0 APPENDICES, DATA, REPORTS, AND FORMS

None

8.0 RECORDS

None

9.0 TOOLS, EQUIPMENT, AND MATERIALS

None10.0 PERSONNEL TRAINING AND CERTIFICATION

None

11.0 FLOW DIAGRAM

None

12.0 RESPONSIBILITIES

Work accomplished within the scope of this organizational instruction will be performed by the Software Assurance representative. The Safety, Reliability, and Quality Assurance Policy and Assessment Department (QD40) may delegate the responsibilities and tasks provided in this organizational instruction to support contractors who are responsible for carrying out the tasks identified herein.